User Events in Cocoa

Session 117

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

User Events in Cocoa

Raleigh Ledet
Cocoa Software Engineer

Overview

Only on Mac OS

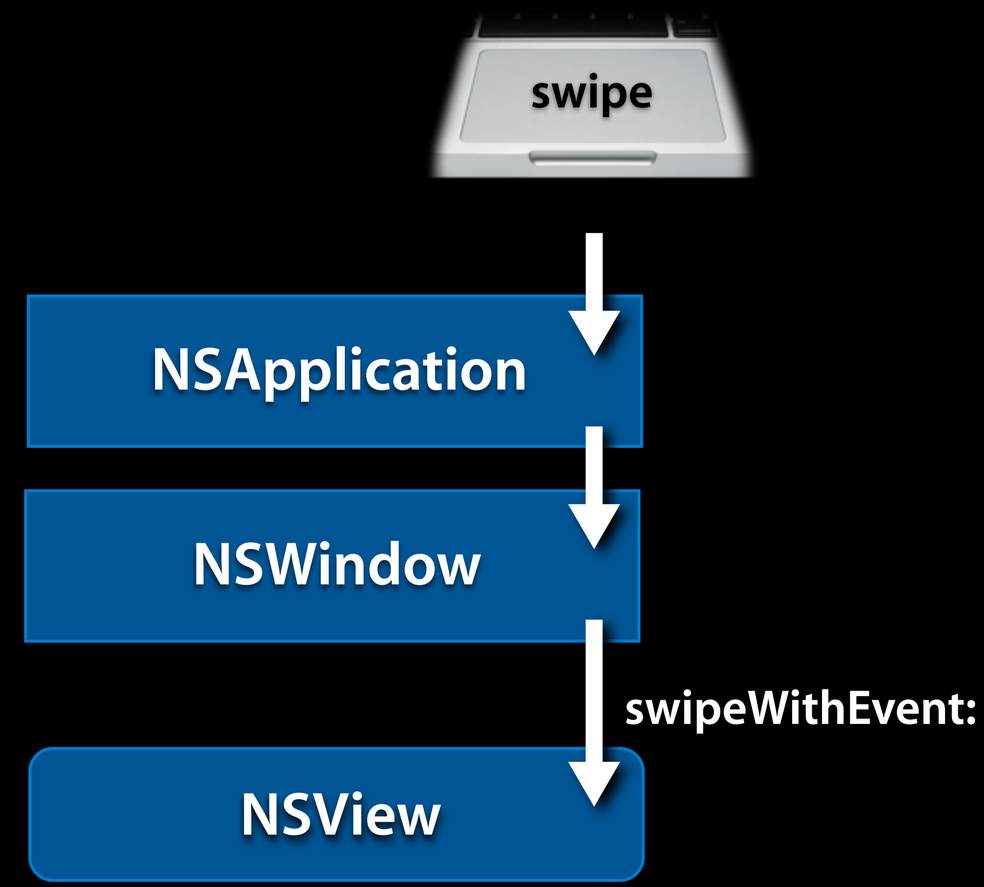
- Event routing and responder chain
- Gesture and multi-touch API
- Event monitoring

Processing Multi-Touch Events on iPhone

Pacific Heights
Thursday 9:00AM

Event Routing

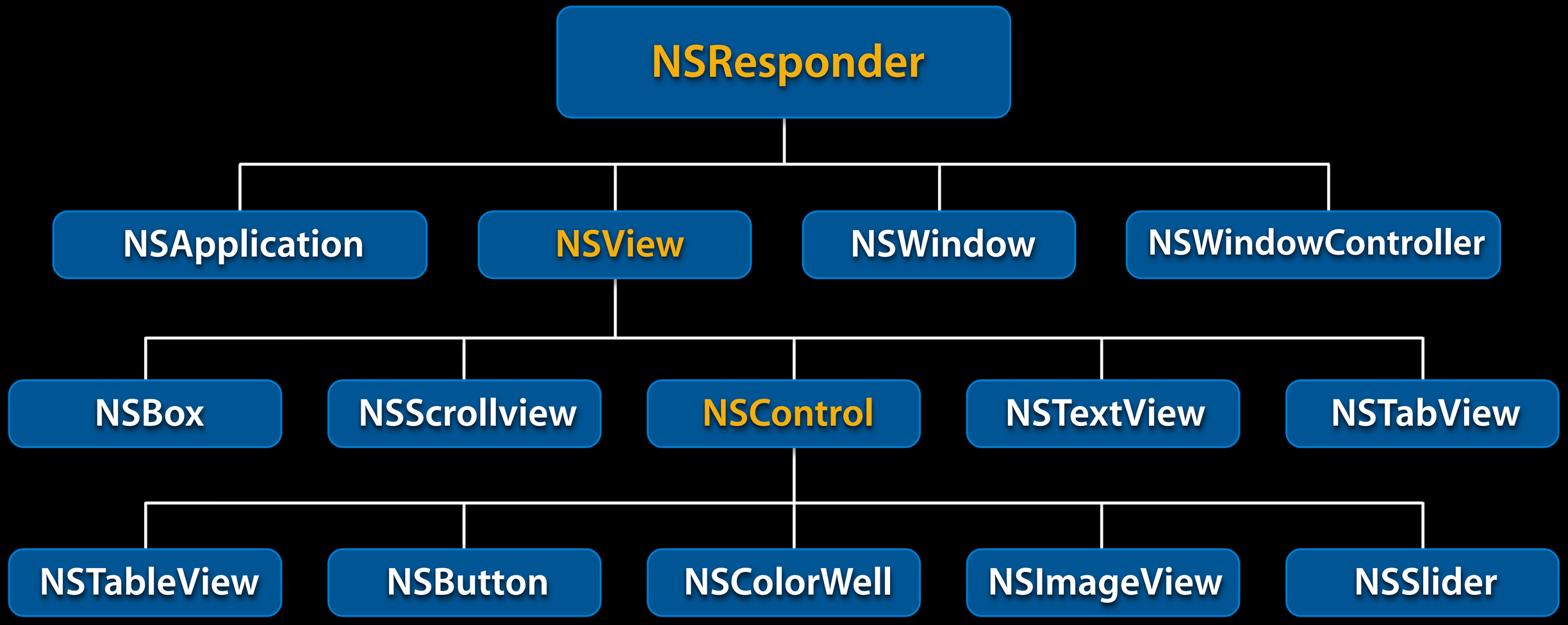
Event Routing



Event Routing

NSView

NSResponder Inheritance



NSResponder

NSResponder

NSResponder


```
mouseDown:(NSEvent *)event
mouseDragged:(NSEvent *)event
mouseUp:(NSEvent *)event
```

```
magnifyWithEvent:(NSEvent *)event
swipeWithEvent:(NSEvent *)event
rotateWithEvent:(NSEvent *)event
```

```
keyDown:(NSEvent *)event
keyUp:(NSEvent *)event
flagsChanged:(NSEvent *)event
```

NSResponder

NSResponder

```
mouseDown:(NSEvent *)event
mouseDragged:(NSEvent *)event
mouseUp:(NSEvent *)event
```

```
magnifyWithEvent:(NSEvent *)event
- (void)swipeWithEvent:(NSEvent *)event;
rotateWithEvent:(NSEvent *)event
```

```
keyDown:(NSEvent *)event
keyUp:(NSEvent *)event
flagsChanged:(NSEvent *)event
```

NSResponder

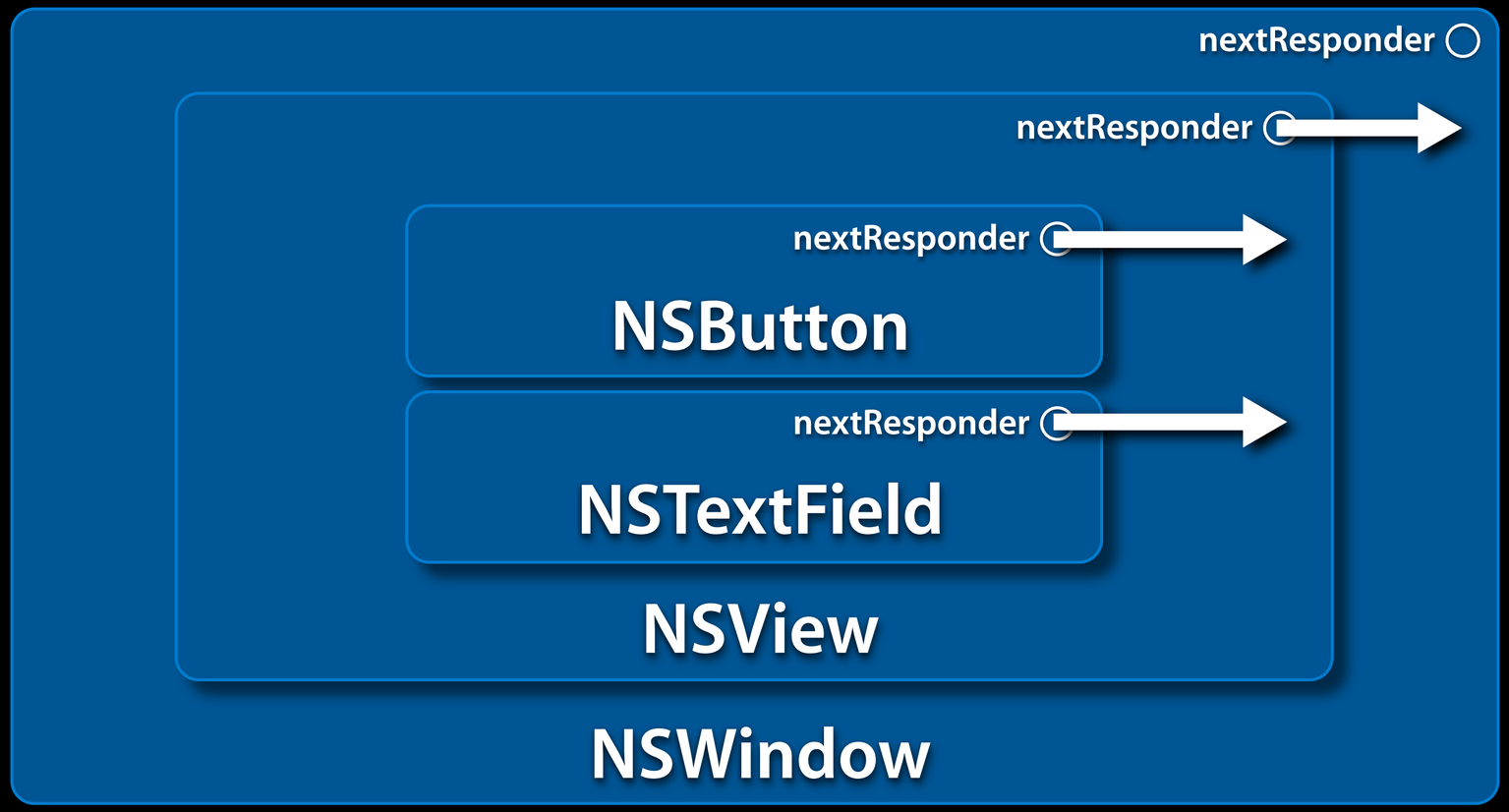
NSResponder

Responder Chain

Responder Chain

Responder Chain

View Hierarchy



Responder Chain

Responder Chain

- Event routing
- Action routing
- Automatic menu enabling
- Error presentation

Responder Chain

- Responder Chain for Action Messages
- Automatic Menu Enabling
- Error Handling Programming Guide for Cocoa

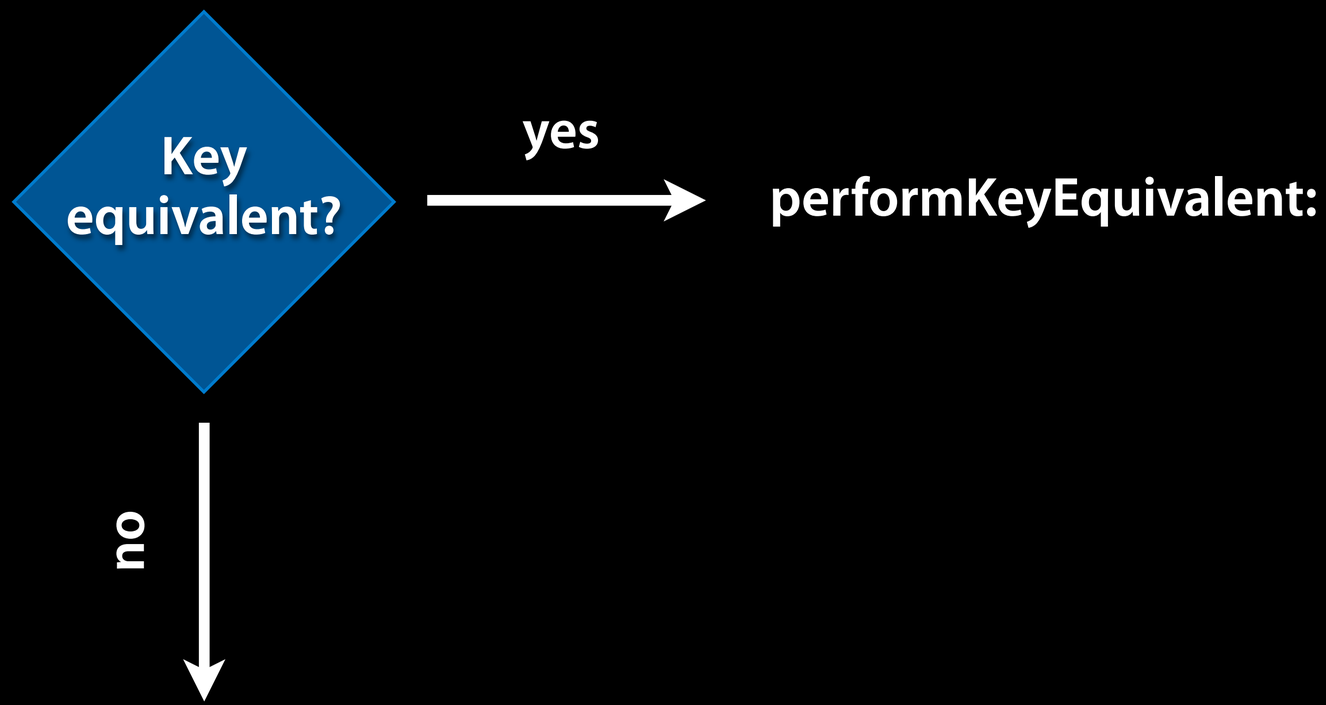
Event Types

Event Types

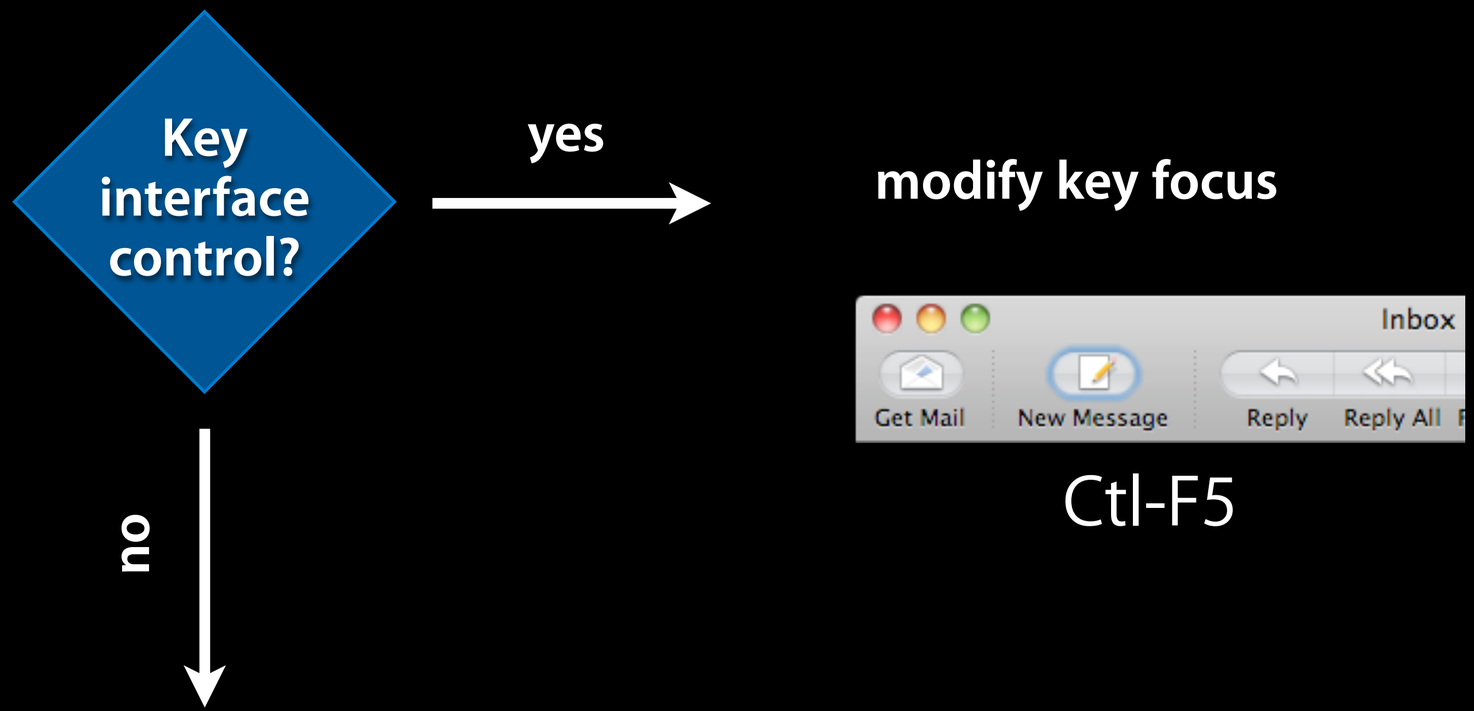
- Keyboard
- Pointer

Keyboard Events

Keyboard Events



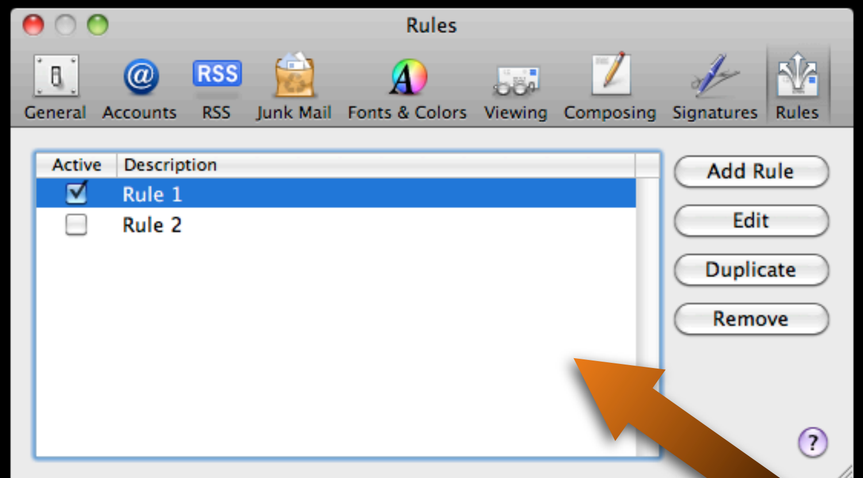
Keyboard Events

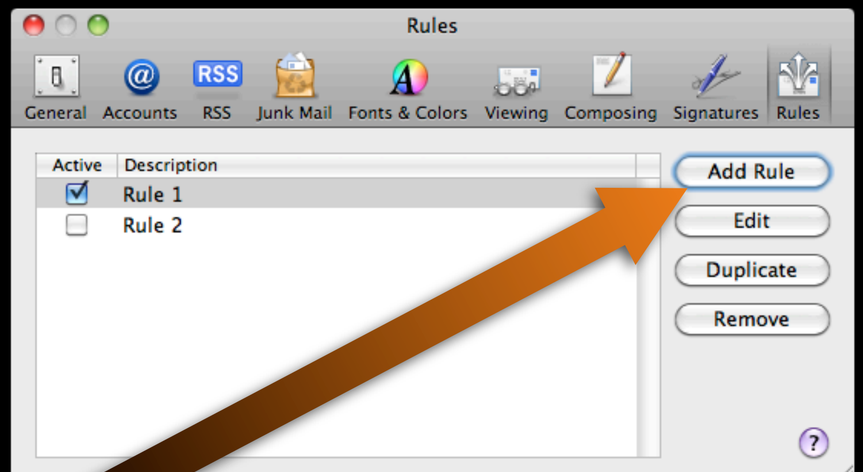


Keyboard Events

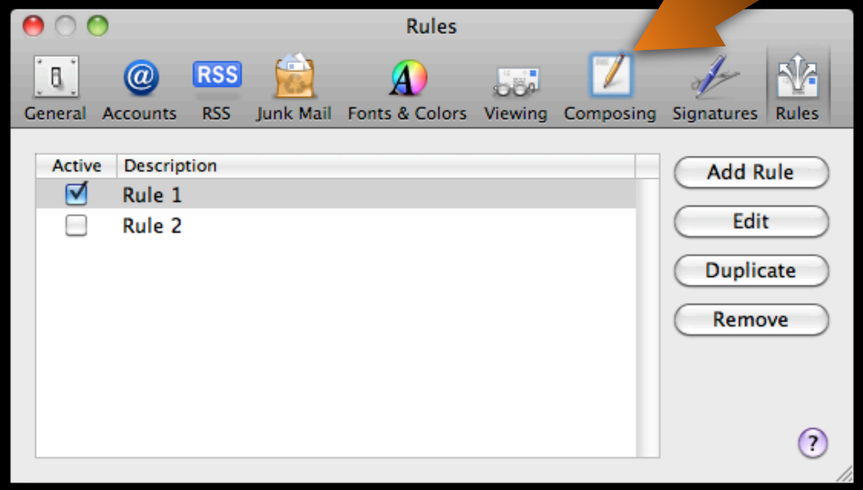
FirstResponder

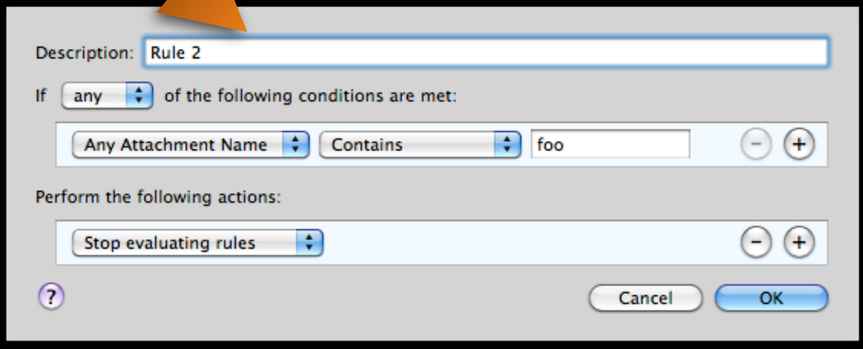
Keyboard Events





FirstResponder

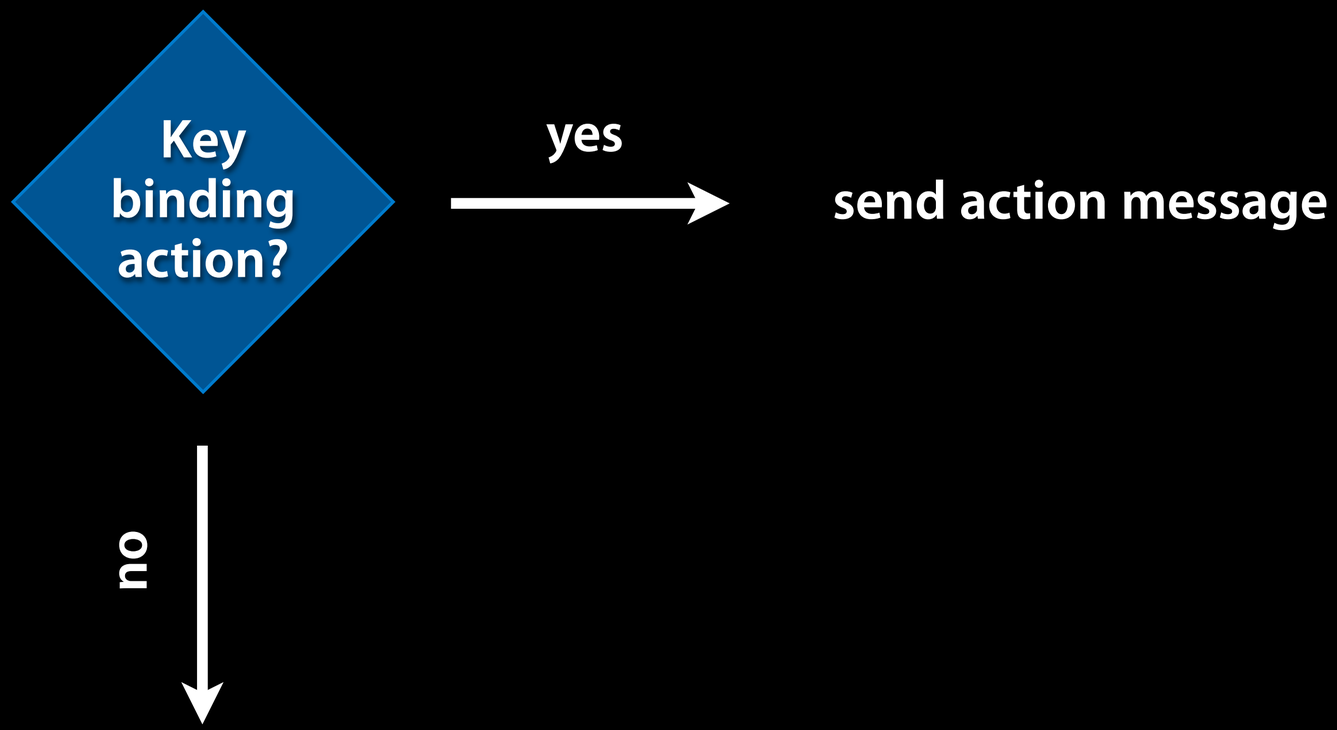




Keyboard Events

```
- (void)keyDown:(NSEvent *)theEvent {
    [self interpretKeyEvents:[NSArray arrayWithObject:theEvent]];
}
```

Keyboard Events



Keyboard Events

```
- (void)insertText:(NSString *)string;
```

Keyboard Events

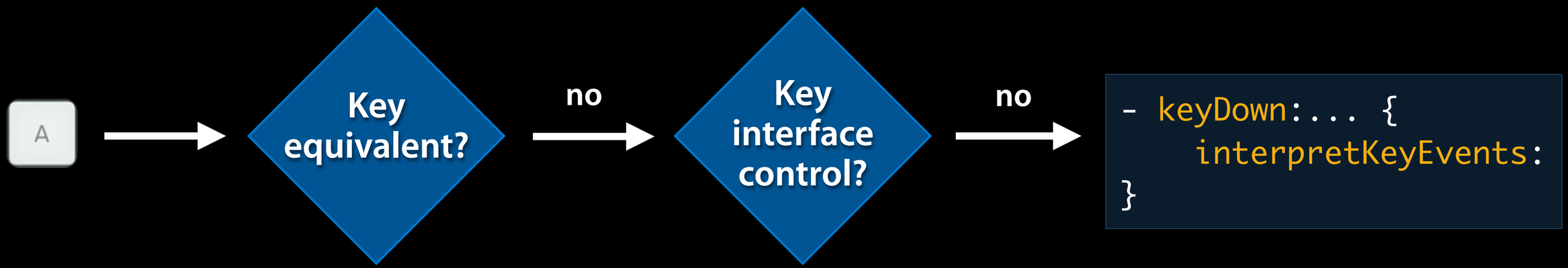
Example

Keyboard Events

Example

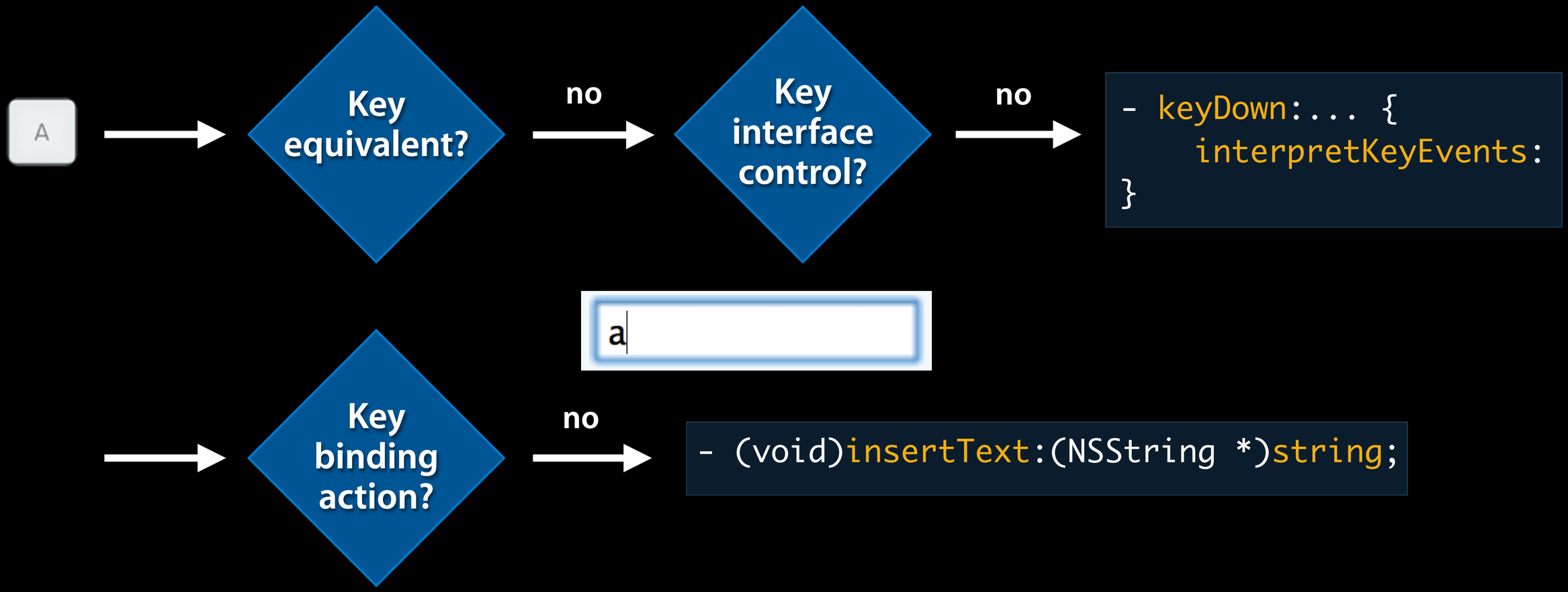
Keyboard Events

Example



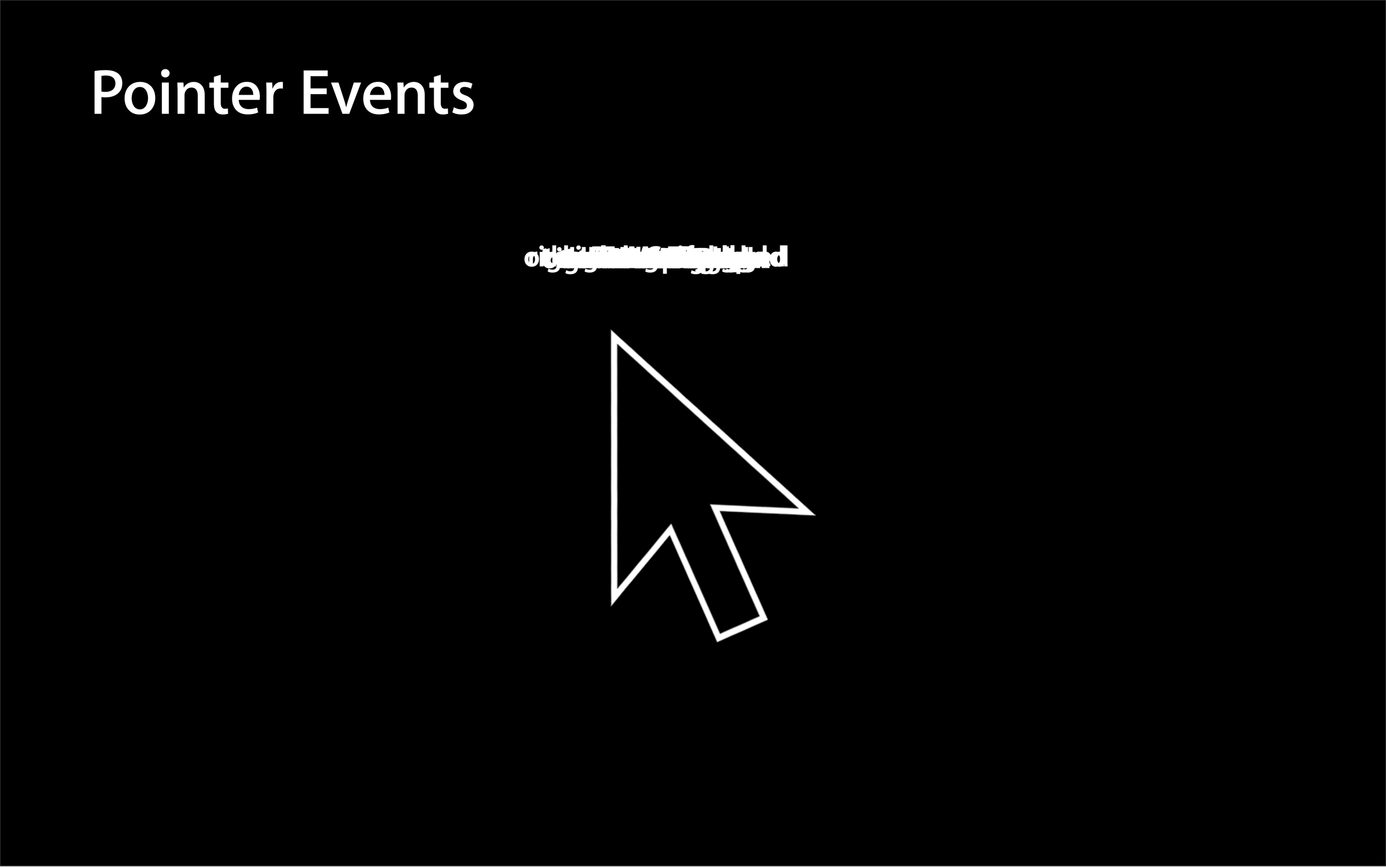
Keyboard Events

Example



Pointer Events

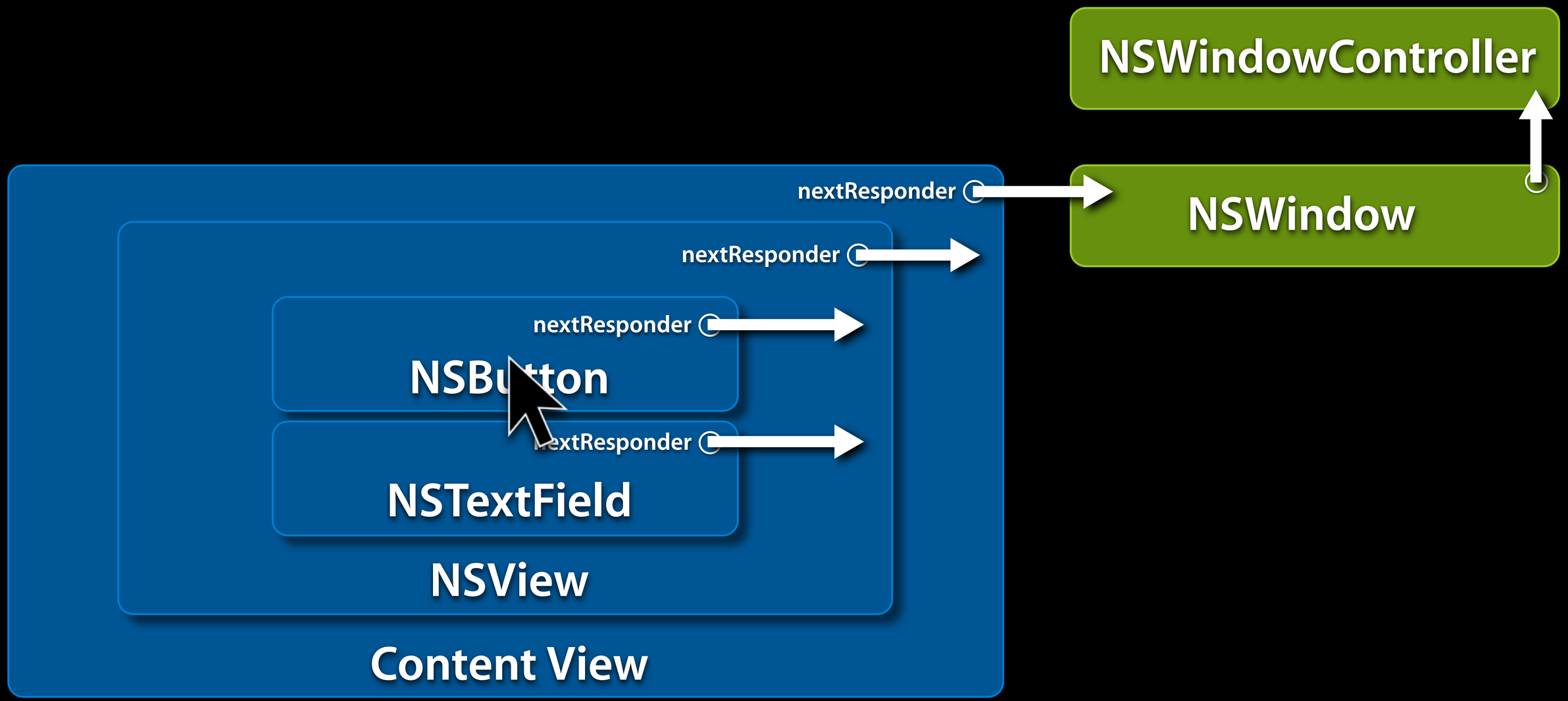
Pointer Events



Demo

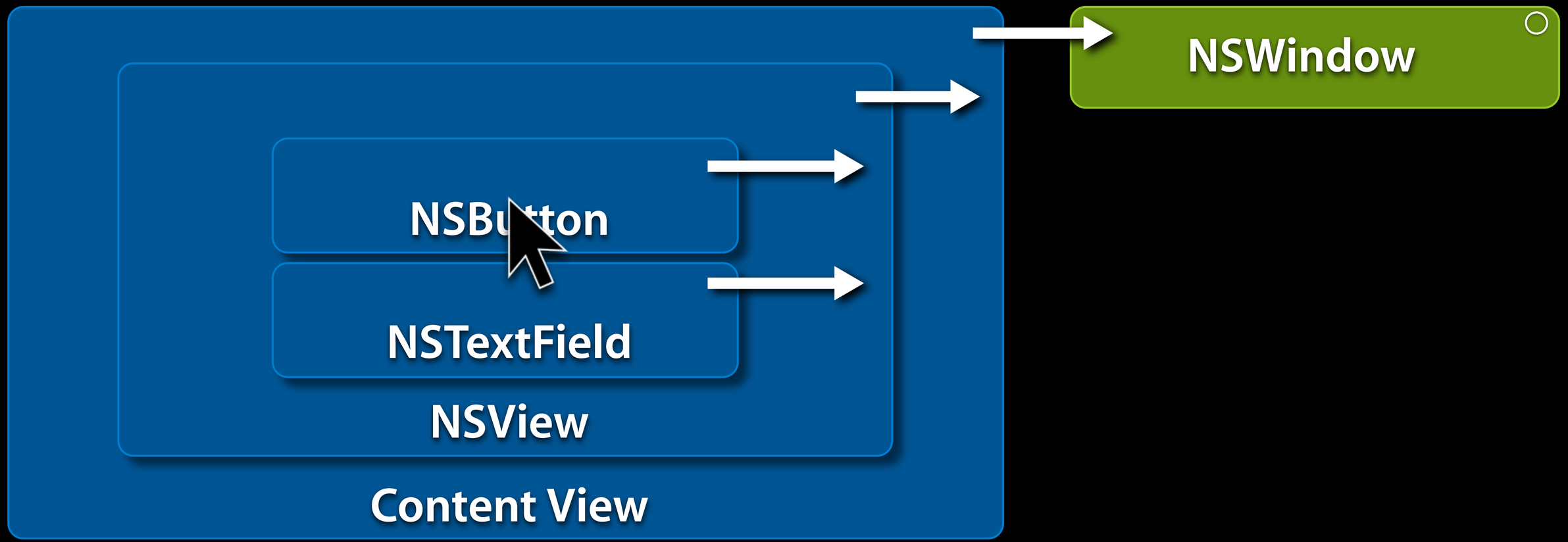
Pointer Events

Event routing



Pointer Events

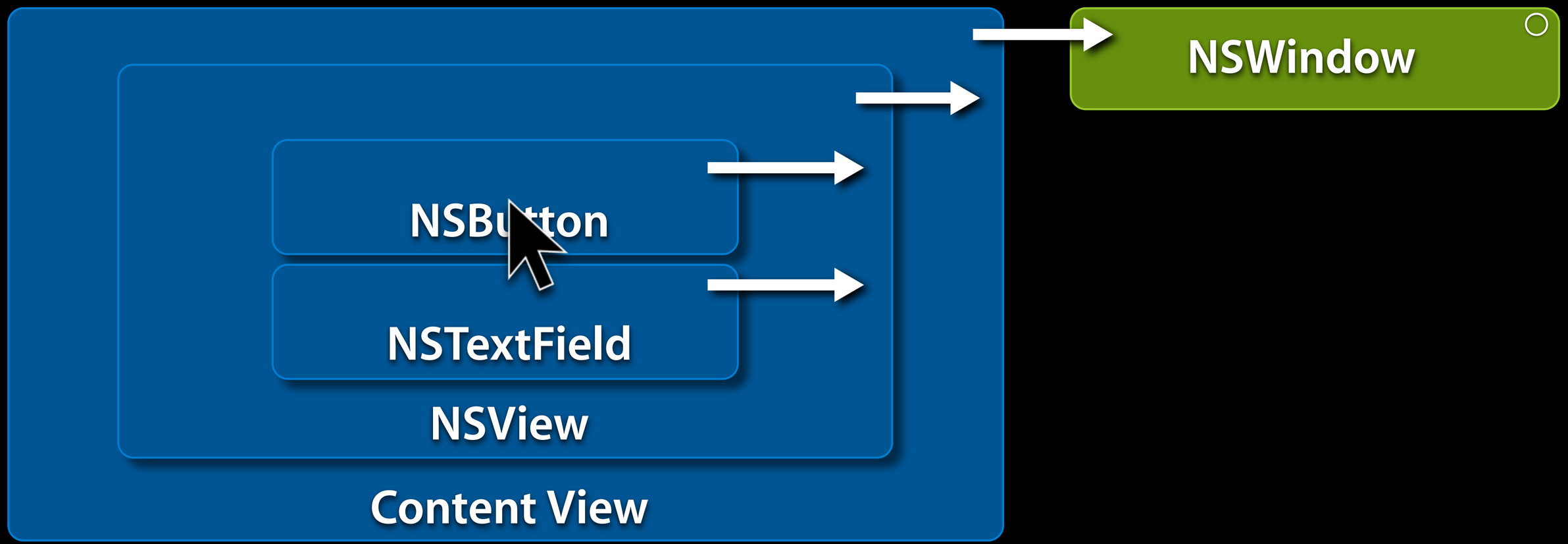
Hit testing



Pointer Events

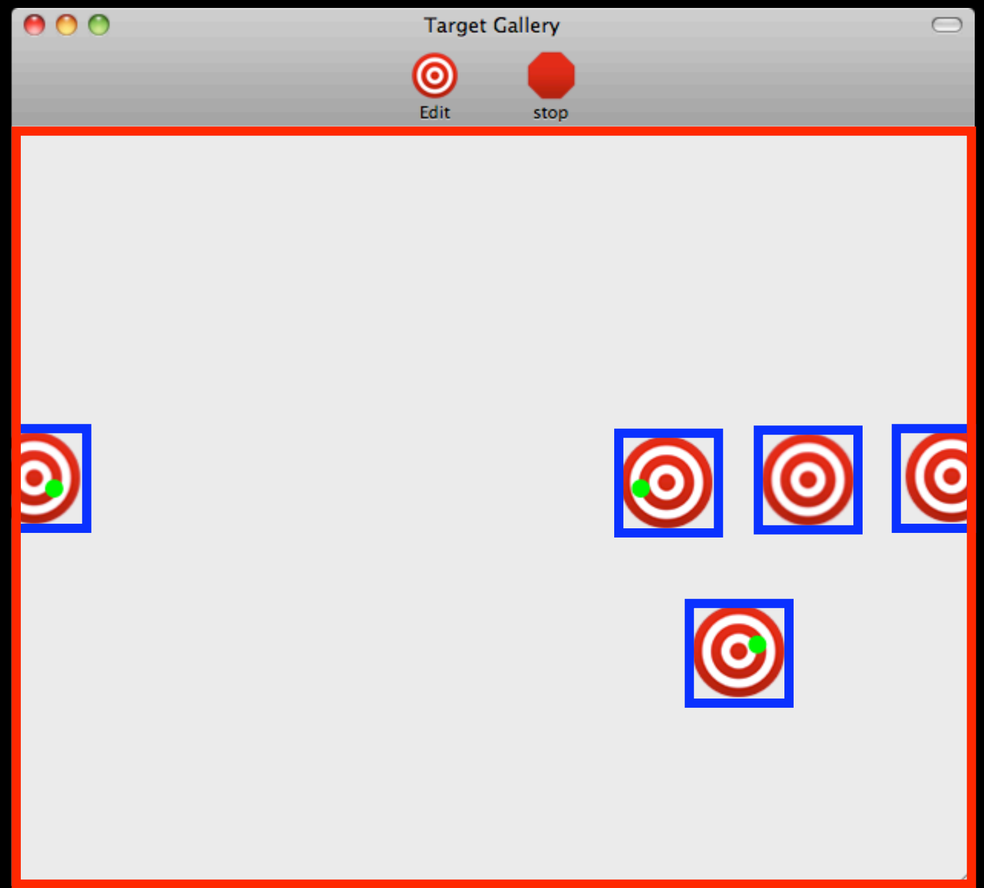
Hit testing

```
-(NSView *)hitTest:(NSPoint)aPoint
```



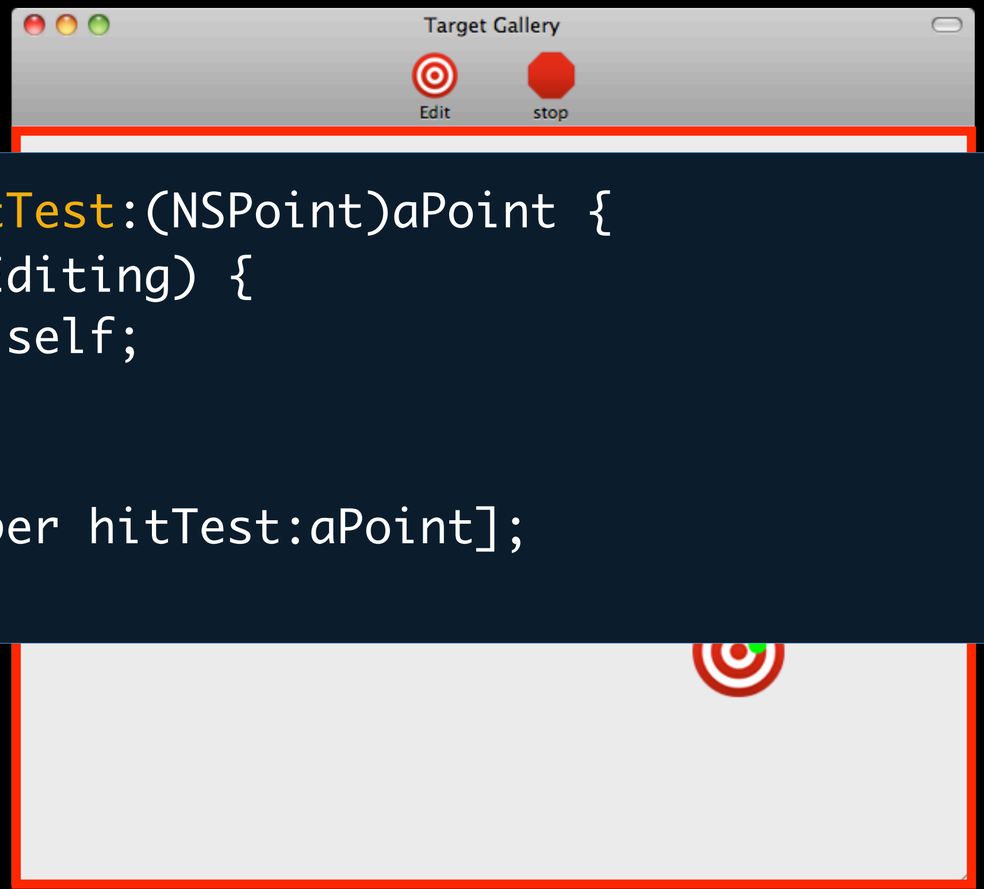
Pointer Events

Hit testing



Pointer Events

Hit testing

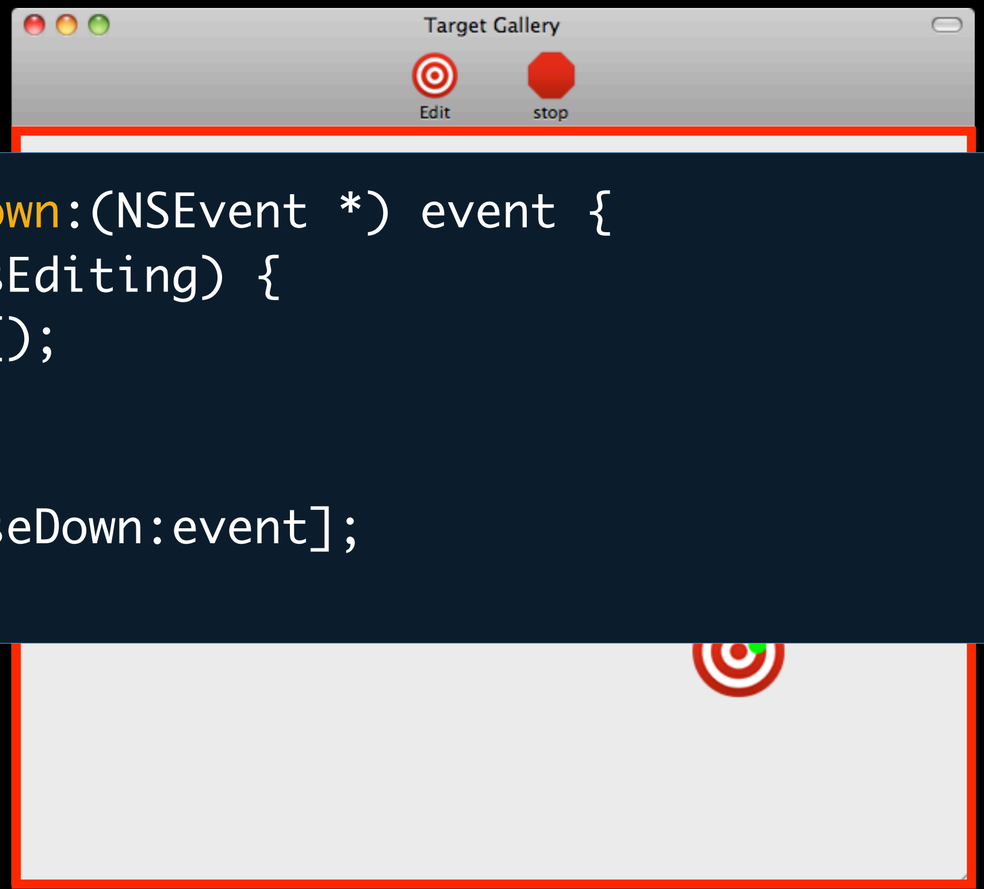



```
- (NSView *)hitTest:(NSPoint)aPoint {
    if(self.isEditing) {
        return self;
    }

    return [super hitTest:aPoint];
}
```

Pointer Events

Hit testing




```
- (void)mouseDown:(NSEvent *) event {
    if(!self.isEditing) {
        NSBeep();
    }

    [super mouseDown:event];
}
```

Mouse Events

Mouse Events

- Subtype
- Best practices
- `delta` vs. `locationInWindow`

Mouse Events

Subtypes

- Mouse
- Tablet
- Touch

```
... :(NSEvent *)event {
    NSInteger subtype = [event subtype];
}
```

Mouse Events

Subtypes

- Mouse
- Tablet
- Touch

Mouse Events

Subtypes

- Mouse
- Tablet
- Touch

deviceID

pressure

tangentialPressure

tilt

rotation

Mouse Events

Subtypes

- Mouse
- Tablet
- Touch

-tabletProximity:

-tabletPoint:

Cocoa Event-Handling Guide: Handling Tablet Events

Mouse Events

Subtypes

- Mouse
- Tablet
- Touch

Mouse Events

Best practices

- Use NSTrackingArea
- Use three-method tracking approach
- Drag thresholds

Mouse Events

Best practices

- Use NSTrackingArea
- Use three-method tracking approach
- Drag thresholds

Mouse Events

Best practices

- Using Tracking-Area Objects
- Use three-method tracking approach
- Drag thresholds

```
- (void)mouseDown:(NSEvent *)theEvent;

- (void)mouseDragged:(NSEvent *)theEvent;

- (void)mouseUp:(NSEvent *)theEvent;
```

Mouse Events

Best practices

- Using Tracking-Area Objects
- The Three-Method Approach
- Drag thresholds

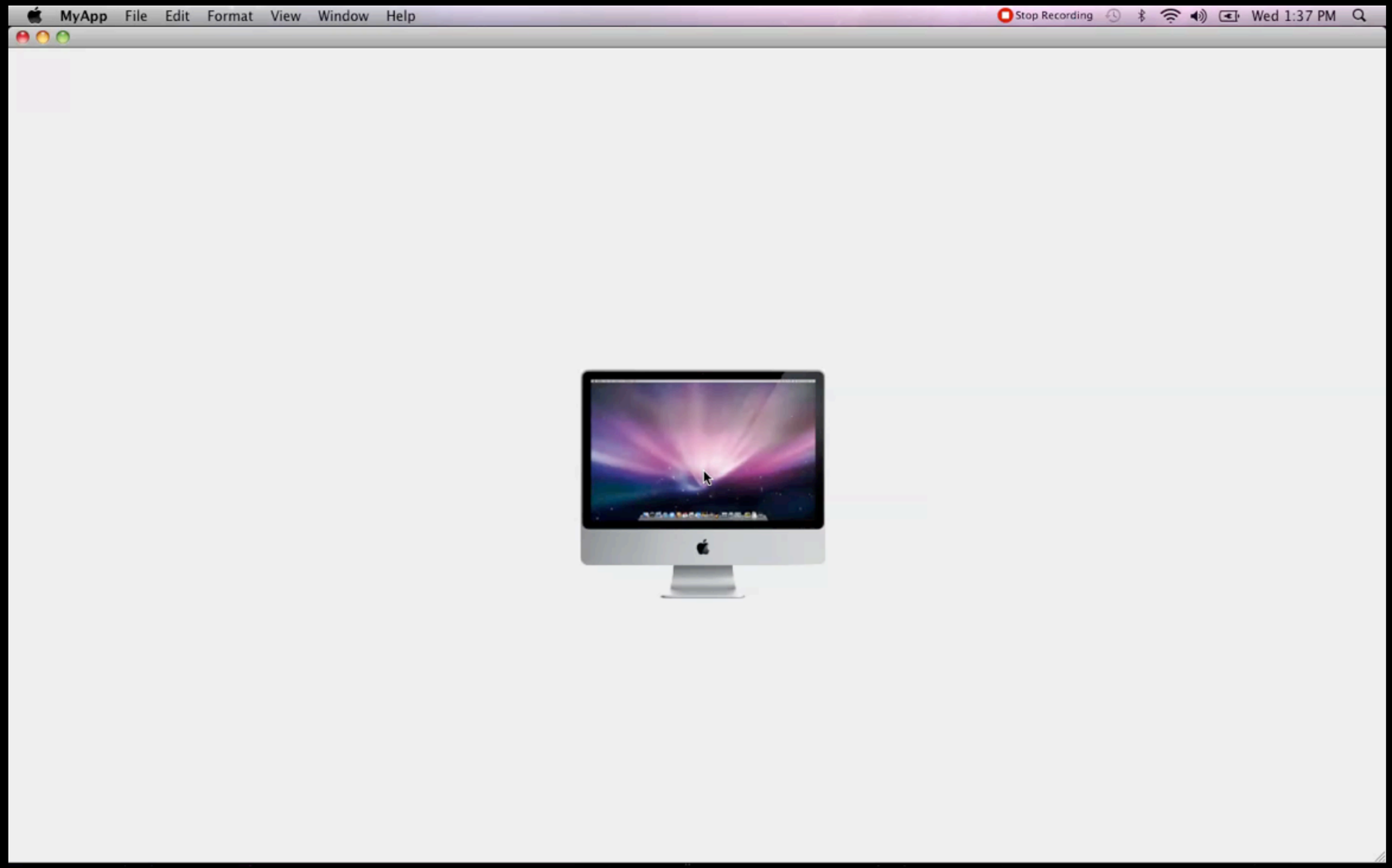
Mouse Events

`delta` vs. `locationInWindow`

```
- (NSPoint)locationInWindow;
- (CGFloat)deltaX;                - (CGFloat)deltaY;
```

Mouse Events

`delta` vs. `locationInWindow`



Mouse Events

delta vs. locationInWindow

- (NSPoint)locationInWindow;

```
- (void)mouseDragged:(NSEvent *)theEvent {
    NSPoint location = [theEvent locationInWindow];
    location = [self convertPoint:location fromView:nil];
    ...
```

Mouse Events

`delta` vs. `locationInWindow`

```
- (NSPoint)locationInWindow;
```

Align with cursor

View localized location of the cursor

Mouse Events

`delta` vs. `locationInWindow`

```
- (NSPoint)locationInWindow;
```

Align with cursor

View localized location of the cursor

```
- (CGFloat)deltaX;
```

Cursor location is irrelevant

Panning

```
- (CGFloat)deltaY;
```

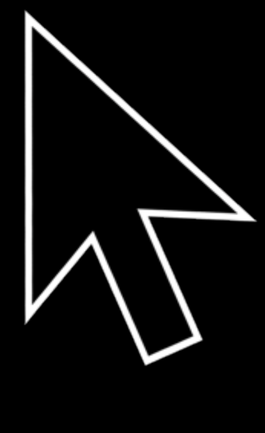
Gestures

Gestures

- Magnify
- Rotate
- Swipe

Gestures

10.5.2



- Magnify
- Rotate
- Swipe

Gestures

- Magnify
- Rotate
- Swipe

```
- (void)magnifyWithEvent:(NSEvent *)event {
    CGFloat change = [event magnification];
}
```

Gestures

- Magnify
- Rotate
- Swipe

```
- (void)rotateWithEvent:(NSEvent *)event {
    float degrees = [event rotation];
}
```

Gestures

- Magnify
- Rotate
- Swipe

```
- (void)swipeWithEvent:(NSEvent *)event {
    NSInteger horizontal = [event deltaX];
    NSInteger vertical   = [event deltaY];
}
```

Gestures

- Magnify
- Rotate
- Swipe

```
Left
[event deltaX] ==  1
```

Gestures

- Magnify
- Rotate
- Swipe

```
Right
[event deltaX] == -1
```

Gestures

- Magnify
- Rotate
- Swipe

Up

```
[event deltaY] ==  1
```

Gestures

- Magnify
- Rotate
- Swipe

Down

```
[event deltaY] == -1
```

Multi-Touch

New

Multi-Touch

Indirect

Direct

Multi-Touch

New

Multi-Touch

New

- Cursor location
- Similar hit testing as mouse
- All touches routed to same view

Demo

Multi-Touch

NSTouch

(NSTouchPhase) phase

(id) identity

(BOOL) isResting

(NSPoint) normalizedPosition

(NSSize) deviceSize

(id) device

Multi-Touch

NSTouch

(NSTouchPhase) phase

(id) identity

(BOOL) isResting

(NSPoint) normalizedPosition

(NSSize) deviceSize

(id) device

Multi-Touch

Touch phases

Began

Moved

Stationary

Ended

Cancelled

Multi-Touch

Touch phases

New

NSTouchPhaseBegan

Moved

Stationary

Ended

Cancelled

Multi-Touch

Touch phases

New

NSTouchPhaseBegan

Moved

Stationary

Ended

Cancelled

Multi-Touch

Touch phases

New

Began

NSTouchPhaseMoved

Stationary

Ended

Cancelled

Multi-Touch

Touch phases

New

NSTouchPhase

- Began
- Moved
- Stationary
- Ended
- Cancelled

Multi-Touch

Touch phases

New

Began

NSTouchPhaseMoved

Stationary

Ended

Cancelled

Multi-Touch

Touch phases

New

Began

Moved

Stationary

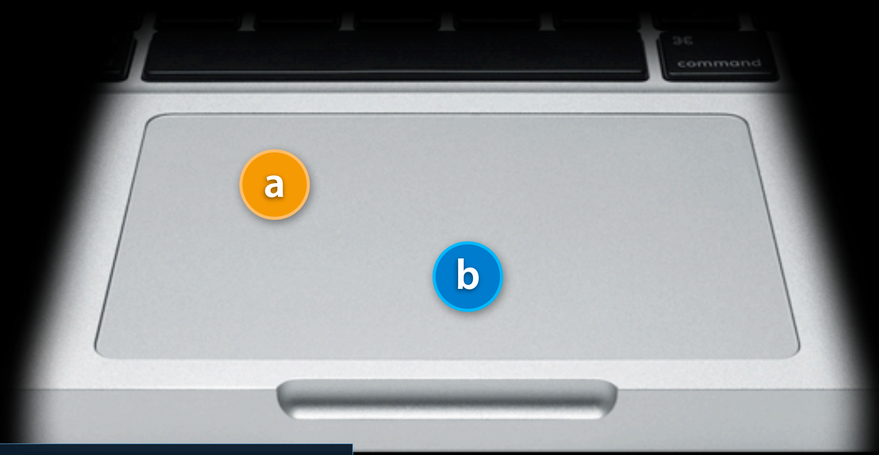
Ended

NSTouchPhaseCancelled

Multi-Touch

NSTouch

New



(NSTouchPhase) phase

(id) identity

```
[touch1.identity isEqual:touch2.identity]
```

(BOOL) isResting

(NSPoint) normalizedPosition

(NSSize) deviceSize

(id) device

Multi-Touch

NSTouch

New

(NSTouchPhase) phase

(id) identity

(BOOL) isResting

(NSPoint) normalizedPosition

(NSSize) deviceSize

(id) device

Multi-Touch

NSTouch

New

(NSTouchPhase) phase

(id) identity

(BOOL) isResting

(NSPoint) normalizedPosition

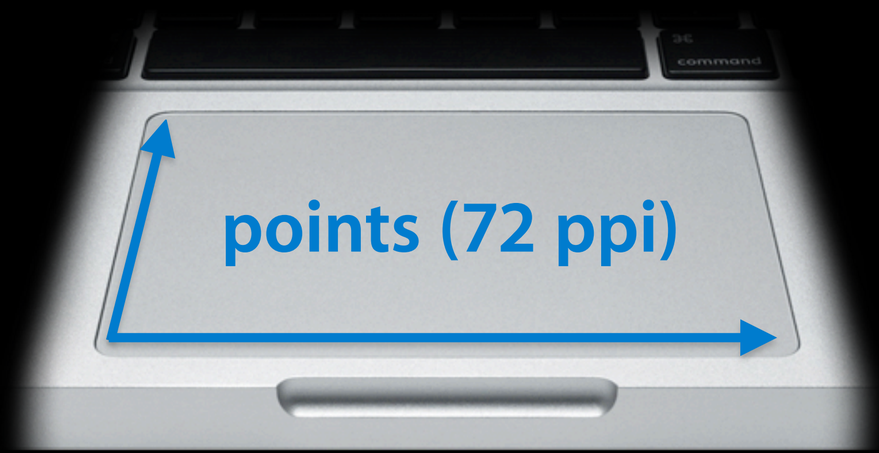
(NSSize) deviceSize

(id) device

Multi-Touch

NSTouch

New



(NSTouchPhase) phase

(id) identity

(BOOL) isResting

(NSPoint) normalizedPosition

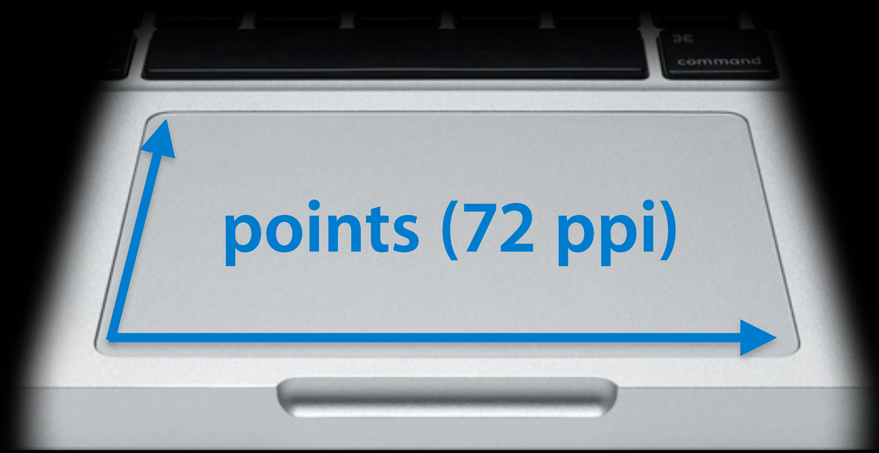
(NSSize) deviceSize

(id) device

Multi-Touch

NSTouch

New



(NSTouchPhase) phase

(id) identity

(BOOL) isResting

(NSPoint) normalizedPosition

(NSSize) deviceSize

(id) device

Multi-Touch

NSTouch

New

(NSTouchPhase) phase

(id) identity

(BOOL) isResting

(NSPoint) normalizedPosition

(NSSize) deviceSize

(id) device

Multi-Touch

Opt-in

```
@interface NSView
...
- (void)setAcceptsTouchEvents:(BOOL)flag;
- (BOOL)acceptsTouchEvents;

- (void)setWantsRestingTouches:(BOOL)flag;
- (BOOL)wantsRestingTouches;
...
@end
```

Multi-Touch

Opt-in

```
@interface NSView
...
- (void)setAcceptsTouchEvents:(BOOL)flag;
- (BOOL)acceptsTouchEvents;

- (void)setWantsRestingTouches:(BOOL)flag;
- (BOOL)wantsRestingTouches;
...
@end
```

Multi-Touch

Opt-in

```
@interface NSView
...
- (void)setAcceptsTouchEvents:(BOOL)flag;
- (BOOL)acceptsTouchEvents;

- (void)setWantsRestingTouches:(BOOL)flag;
- (BOOL)wantsRestingTouches;
...
@end
```

Multi-Touch

NSResponder methods

```
- (void)touchesBeganWithEvent:(NSEvent *)event
- (void)touchesMovedWithEvent:(NSEvent *)event
- (void)touchesEndedWithEvent:(NSEvent *)event
- (void)touchesCancelledWithEvent:(NSEvent *)event
```

Multi-Touch

NSEvent methods

```
- (NSSet *)touchesMatchingPhase:(NSTouchPhase)phase inView:(NSView *)view
```

Multi-Touch

NSEvent methods

…MatchingPhase:

NSTouchPhaseMoved | NSTouchPhaseStationary

Multi-Touch

NSEvent methods

…MatchingPhase:

NSTouchPhaseTouching = Began | Moved | Stationary

Multi-Touch

NSEvent methods

…MatchingPhase:

NSTouchPhaseTouching = Began | Moved | Stationary

NSTouchPhaseAny

Multi-Touch

Sample code

```
- (void)touchesBeganWithEvent:(NSEvent *)event {
    NSSet *touches = [event
                      touchesMatchingPhase:NSTouchPhaseTouching
                      inView:self.view];

    if(touches.count == 2) {
        // Setup for 2 touch tracking
    } else if(touches.count > 2) {
        // More than 2 touches. Only track 2.
        [self cancelTracking];
    }
}
```

Multi-Touch

Sample code

New

```
- (void)touchesBeganWithEvent:(NSEvent *)event {
    NSSet *touches = [event
                      touchesMatchingPhase:NSTouchPhaseTouching
                      inView:self.view];

    if(touches.count == 2) {
        // Setup for 2 touch tracking
    } else if(touches.count > 2) {
        // More than 2 touches. Only track 2.
        [self cancelTracking];
    }
}
```

Multi-Touch

Sample code

```
- (void)touchesBeganWithEvent:(NSEvent *)event {
    NSSet *touches = [event
                      touchesMatchingPhase:NSTouchPhaseTouching
                      inView:self.view];

    if(touches.count == 2) {
        // Setup for 2 touch tracking
    } else if(touches.count > 2) {
        // More than 2 touches. Only track 2.
        [self cancelTracking];
    }
}
```

Multi-Touch

Sample code

```
- (void)touchesBeganWithEvent:(NSEvent *)event {
    NSSet *touches = [event
                      touchesMatchingPhase:NSTouchPhaseTouching
                      inView:self.view];

    if(touches.count == 2) {
        // Setup for 2 touch tracking
    } else if(touches.count > 2) {
        // More than 2 touches. Only track 2.
        [self cancelTracking];
    }
}
```

Multi-Touch

Best practices

- Selectively ignore events
- Consider tracking thresholds
- Forward gesture events to nextResponder

Multi-Touch

Concurrent events

Touch

Touch
Mouse

Touch
Gesture

Multi-Touch

Best practices

- Selectively ignore events
- Consider tracking thresholds
- Forward gesture events to nextResponder

Multi-Touch

Best practices

- Selectively ignore events
- Consider tracking thresholds
- Forward gesture events to nextResponder

Multi-Touch

Best practices

- Selectively ignore events
- Consider tracking thresholds
- Forward gesture events to nextResponder

Multi-Touch vs. Gestures

touchesBegan
touchesMoved
touchesEnded
touchesCancelled

vs.

Magnify
Rotate
Swipe

Event Big Brother

Funnel Points

```
-(void)sendEvent:(NSEvent *)theEvent
```

Event Monitoring

- Local monitors
- Global monitors

Event Monitoring

Local

```
@interface NSEvent ...
+ (id)addLocalMonitorForEventsMatchingMask:(NSEventMask)mask
       handler:(NSEvent* (^)(NSEvent*))block;
```

- Events that are targeted for your app
- Can modify the event
- No events from tracking loop
- Hold on to returned object

Event Monitoring

Local

```
@interface NSEvent ...
+ (id)addLocalMonitorForEventsMatchingMask:(NSEventMask)mask
       handler:(NSEvent* (^)(NSEvent*))block;
```

- Events that are targeted for your app
- Can modify the event
- No events from tracking loop
- Hold on to returned object

Event Monitoring

Local

```
@interface NSEvent ...
+ (id)addLocalMonitorForEventsMatchingMask:(NSEventMask)mask
       handler:(NSEvent* (^)(NSEvent*))block;
```

- Events that are targeted for your app
- Can modify the event
- No events from tracking loop
- Hold on to returned object

Event Monitoring

Local

```
@interface NSEvent ...
+ (id)addLocalMonitorForEventsMatchingMask:(NSEventMask)mask
       handler:(NSEvent* (^)(NSEvent*))block;
```

- Events that are targeted for your app
- Can modify the event
- No events from tracking loop
- Hold on to returned object

Event Monitoring

Global

```
@interface NSEvent ...
+ (id)addGlobalMonitorForEventsMatchingMask:(NSEventMask)mask
       handler:(void (^)(NSEvent*))block;
```

- No events already targeted to your application
- Observe only
- No key events
- Hold on to returned object

Event Monitoring

Global

```
@interface NSEvent ...
+ (id)addGlobalMonitorForEventsMatchingMask:(NSEventMask)mask
       handler:(void (^)(NSEvent*))block;
```

- No events already targeted to your application
- Observe only
- No key events
- Hold on to returned object

Event Monitoring

Removing event monitors

- Cocoa manages event monitors
- Event monitors must explicitly be removed

```
@interface NSEvent ...
+ (void)removeMonitor:(id)eventMonitor;
```

Demo

Event Monitoring

A word of caution

New

Potential performance impact

Sample Code for This Session

User events in Cocoa

- Download the sample code
  - “TargetGallery”
  - “LightTable”

Materials available at:
http://developer.apple.com/wwdc/attendee/

Documentation

Cocoa Event-Handling Guide: Event Architecture

http://developer.apple.com/documentation/Cocoa/Conceptual/EventOverview/EventArchitecture/EventArchitecture.html

Application Menu and Pop-up List Programming Topics for Cocoa

http://developer.apple.com/documentation/Cocoa/Conceptual/MenuList/Articles/EnablingMenuItems.html

Error Handling Programming Guide for Cocoa

http://developer.apple.com/documentation/Cocoa/Conceptual/ErrorHandlingCocoa/ErrorHandling/ErrorHandling.html

Cocoa Event-Handling Guide: Using Tracking-Area Objects

http://devworld.apple.com/documentation/Cocoa/Conceptual/EventOverview/TrackingAreaObjects/chapter_7_section_1.html

Documentation

Cocoa Event-Handling Guide: Handling Mouse Events

http://developer.apple.com/documentation/Cocoa/Conceptual/EventOverview/HandlingMouseEvents/HandlingMouseEvents.html

Cocoa Release Notes

Q&A

WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09